

# **TDMQ for RabbitMQ**

## **Getting Started**

### **Product Documentation**



## Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

## Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

## Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

# Contents

## Getting Started

Resource Creation and Preparation

Using SDK to Send/Receive Message

# Getting Started

## Resource Creation and Preparation

Last updated : 2024-06-26 15:56:25

### Overview

This document describes how to create an exclusive cluster in the TDMQ console and create resources such as vhosts, exchanges, and queues in the open-source RabbitMQ console. This helps you prepare the resources required to run a client.

### Directions

#### Step 1: Create a cluster.

1. Log in to [RabbitMQ Console](#).
2. Choose **Cluster Management** > **Cluster List** in the left sidebar, click **Create Cluster** to proceed to the purchase page.
3. On the purchase page, select the target instance specification and click **Buy Now** to complete the creation.
4. Click the cluster ID to enter the **Basic Info** page and get the server connection information in the **Client Access** section.

Client Access ⓘ					<a href="#">Add a routing policy</a>
Access Type	Access Policy	Public ...	Network	Operat...	
VPC Network	-	-	<a href="#">vpc-fs6qq7yn</a> ⓘ <a href="#">subnet-8ah6a7rs</a> ⓘ amqp://10... ⓘ	Delete	

#### Step 2: Create a Vhost.

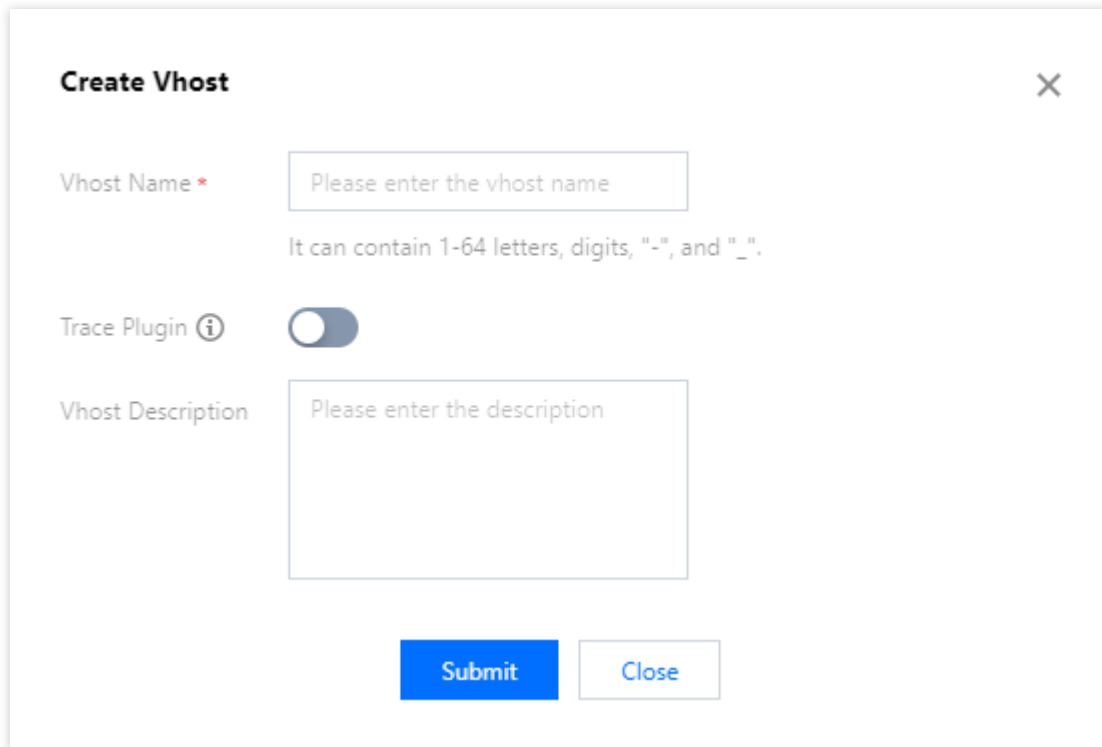
1. Click the ID of the cluster just created to enter the **Basic Info** page.
2. Select the **Vhost** tab at the top and click **Create** to enter the **Create Vhost** page.

3. In the **Create Vhost** window, configure the vhost attributes:

Vhost Name: Enter the vhost name, which cannot be modified after creation and can contain 1–64 letters, digits, "-", and "\_".

Remarks: Enter the vhost remarks.

4. Click **Submit**.



**Create Vhost**

Vhost Name \*

It can contain 1-64 letters, digits, "-", and "\_".

Trace Plugin ☐

Vhost Description

**Submit** **Close**

### Step 3: Create a user and grant permissions.

Every cluster has a user named "admin" by default. You can configure permissions for this default user or create new users as needed.

1. Select the **User and Permission** tab at the top of the page and click **Create User** on the **User Management** tab.
2. Enter the username and password and click **Submit**.

### Create User

Username \*

Please enter the name

This field is required. Please enter 1-64 letters, digits, or symbols ("-" or "\_").

Password \*

Please enter user password

This field is required and must contain 8-64 characters in at least two of the following types: lowercase letters, uppercase letters, digits, and symbols ([~!@#\$%^&\*\_{}|~;.,?/).  
Please keep your password properly and remember it.

Confirm Password \*

Please enter the user password again

This field is required and must contain 8-64 characters in at least two of the following types: lowercase letters, uppercase letters, digits, and symbols ([~!@#\$%^&\*\_{}|~;.,?/).

Role

administrator

For permission description for different roles, see [Documentation](#).

Description

Please enter the description

Submit

Close

3. On the **User and Permission** tab, select the **Permission List** tab and click **Configure Permission**.

4. On the **Configure Permission** page, select the target vhost and user and set permission rules.

Permission rules can match resources through **regex**. For example, if you select **Configuration** and enter "test.\*" in the input box, then the user will be granted the permission to configure all resources with a name starting with "test-" under the current vhost.

5. Click **Submit**.

### Configure Permission ✕

Vhost \* (AMQP default vhost) ▼

No vhost is available? Please go to the [Vhost](#) tab to create one.

Username \* admin ▼

Permission ⓘ

☒ Configuration

If this option is selected, the defa

☒ read

If this option is selected, the defa

☐ write

If this option is selected, the defa

For more permission type information, see [here](#)

Submit

Close

#### Step 4: Create an Exchange.

1. In the left sidebar, choose **Cluster Management** > **Exchange** , choose the recently created cluster and Vhost, and click **Create** .
2. Enter the Exchange Name, select the Route Type, and optionally fill in other Advanced Parameters.
3. Click **Submit**.

### Create Exchange

Current Vhost(AMQP default vhost)

Exchange Name \*

Please enter the name

This field is required. Please enter 1-64 letters, digits, or symbols (".", "-", or "\_").

Route Type \*

Please select

For route type descriptions, see [Route Type](#)

Durable

☒

If this option is enabled, the exchange will still exist after the service is restarted; if it is disabled, the exchange will disappear after the service restart and needs to be created again.

AutoDelete

☐

If this option is enabled, the exchange will be automatically deleted when the last queue bound to it is deleted.

Internal

☐

If this option is enabled, this exchange cannot be directly used by producers but bound with other exchanges.

Exchange Description

Please enter the description

Up to 128 characters

Advanced Settings ▶

Submit

Close

## Step 5: Create a Queue.

1. In the Left Navigation Bar, select **Cluster > Queue**, choose the recently created cluster and vhost, and click **Create**.
2. Enter the Queue Name, select the Queue Type, Node, and optionally fill in Common Parameters and Other Advanced Options.
3. In the final step, click **Next** to complete the Queue creation.



### Create Queue

1 Basic Info

2 Common Parameters

3 Other Advanced Options

Current Vhost

(AMQP default vhost)

Queue Name \*

Please enter the name

This field is required. Please enter 1-64 letters, digits, or symbols (" ", "-", or "\_").

Type

Regular queue

Durable

☒

Node

rabbit@rabbitmq-broker-0.rabbitmq-broker-internal.amqp-25mpxb9x.svc

AutoDelete

☐

The queue will be immediately deleted after the last consumer unsubscribes from it.

Queue Description

Please enter the description

Up to 128 characters

Next

Close

## Step 6: Bind the routing relationship.

1. On the Vhost List page, enter the ID of the newly created Vhost to access the Basic Info page.

2. On the Top of the page, select the Routing Relationship tab, and click **Create**.

Choose the recently created Exchange as the Source Exchange, enter the Binding Key, select Queue as the Binding Target Type, and choose the newly created Queue as the Binding Target.

3. Click **Submit** to complete the binding.

**Create Binding** ×

Current Vhost

(AMQP default vhost)

Source Exchange \*

Binding Key \*

It can only contain 1-255 letters, digits, and symbols (-\_@#\*).

Binding Target Type

Exchange

Queue

Binding Target \*

Submit

Close

# Using SDK to Send/Receive Message

Last updated : 2024-06-26 11:15:09

## Overview

This document describes how to use open-source SDK to send and receive messages by using the SDK for Java as an example and helps you better understand the message sending and receiving processes.

## Prerequisites

You have created the required resources as instructed in [Resource Creation and Preparation](#).

You have installed [JDK 1.8 or later](#).

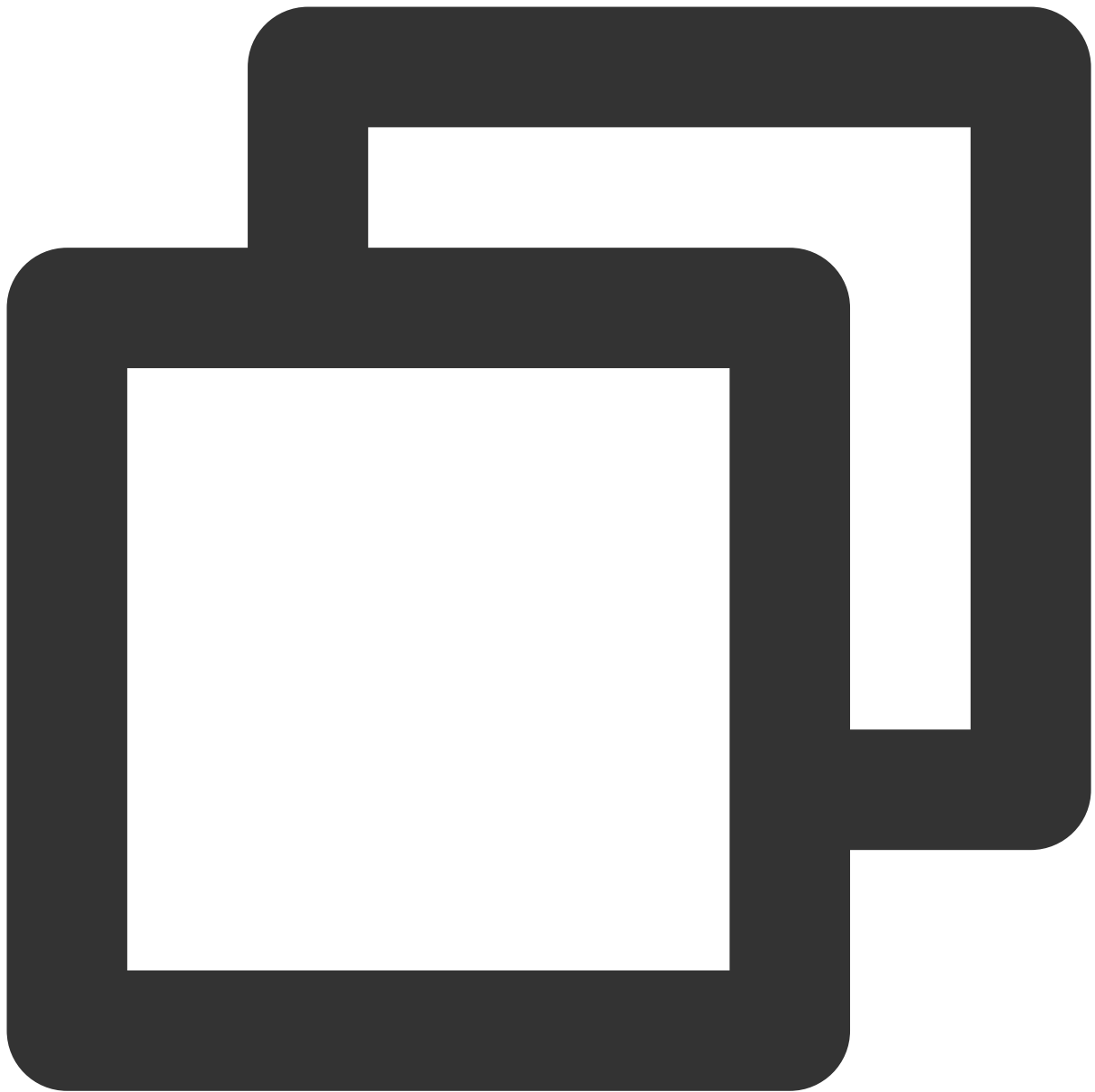
You have installed [Maven 2.5 or later](#).

You have downloaded the [demo](#).

## Directions

### Step 1. Install the Java dependency library

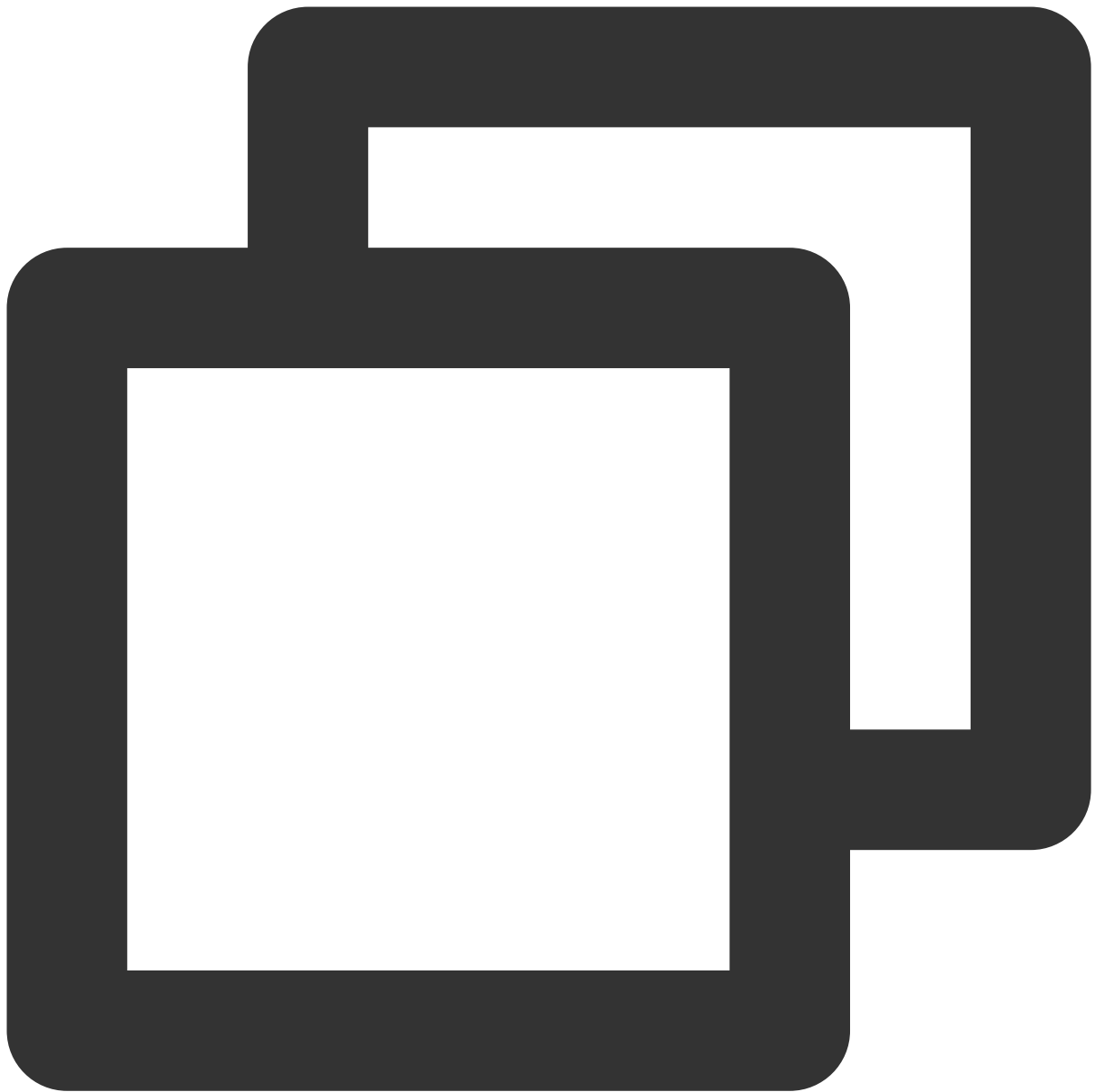
Add the following dependencies to the `pom.xml` file:



```
<!-- in your <dependencies> block -->
<dependency>
  <groupId>com.rabbitmq</groupId>
  <artifactId>amqp-client</artifactId>
  <version>5.13.0</version>
</dependency>
```

## Step 2. Produce messages

Compile and run `MessageProducer.java` .



```
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import com.tencent.tdmq.demo.cloud.Constant;

/**
 * Message producer
 */
public class MessageProducer {
```

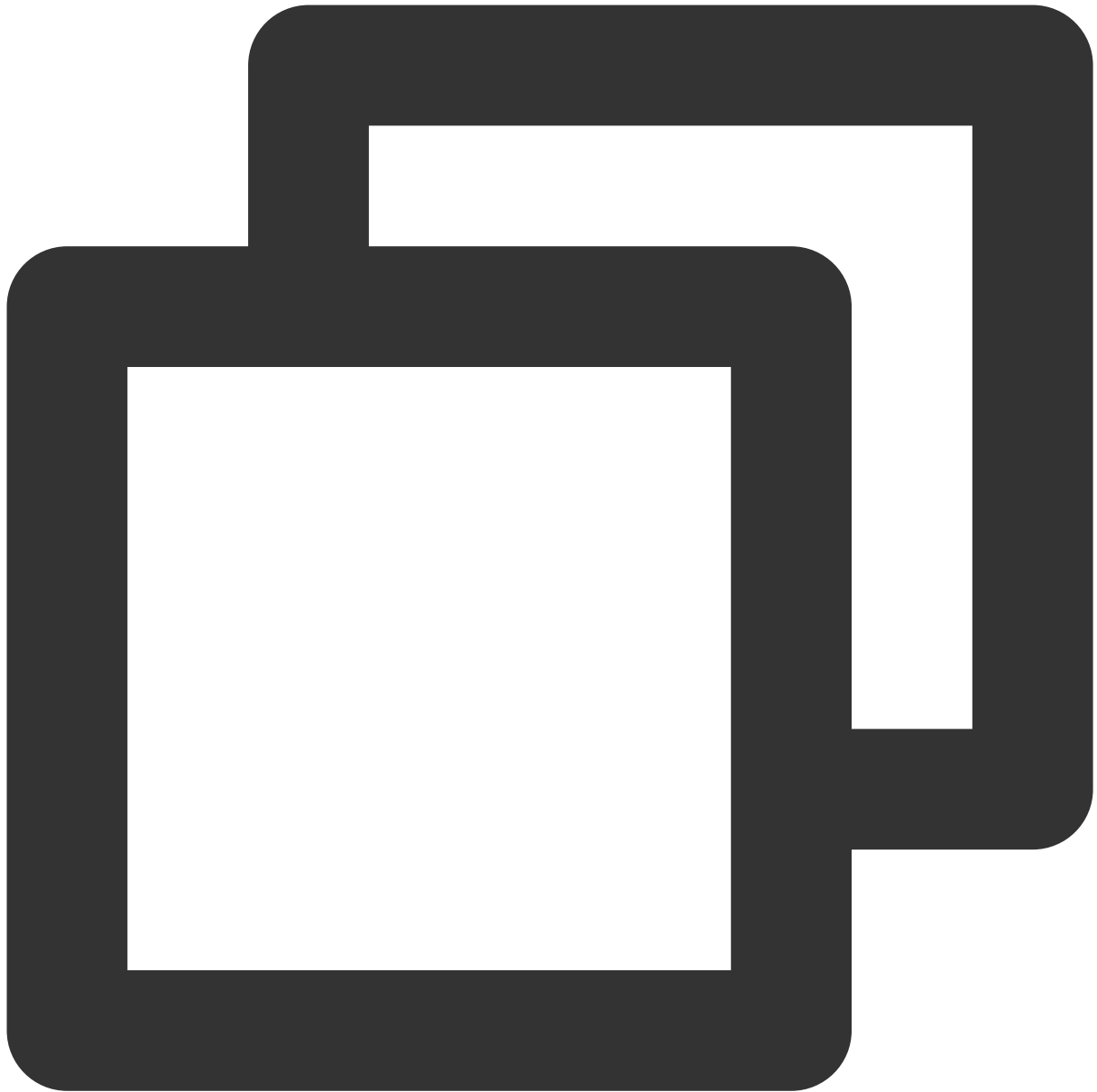
```
/**  
  
 * Exchange name  
 */  
private static final String EXCHANGE_NAME = "exchange_name";  
  
public static void main(String[] args) throws Exception {  
    // Connection factory  
    ConnectionFactory factory = new ConnectionFactory();  
    // Set the service address (replace with the access point address copied in  
    factory.setUri("amqp://***");  
    // Set the vhost (copy the vhost name in the open-source RabbitMQ console)  
    factory.setVirtualHost(VHOST_NAME);  
    // Set the username (use the username in the permission configuration of the  
    factory.setUsername(USERNAME);  
    // Set the password (use the user key)  
    factory.setPassword("****");  
    // Get the connection address and establish the channel  
    try (Connection connection = factory.newConnection(); Channel channel = conn  
        // Bind the message exchange (`EXCHANGE_NAME` must exist in the TDMQ for  
        channel.exchangeDeclare(EXCHANGE_NAME, "fanout");  
        for (int i = 0; i < 10; i++) {  
            String message = "this is rabbitmq message " + i;  
            // Publish a message to the exchange, which will automatically deliv  
            channel.basicPublish(EXCHANGE_NAME, "", null, message.getBytes());  
            System.out.println(" [producer] Sent '" + message + "'");  
        }  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

Parameter	Description
EXCHANGE_NAME	Exchange name, which can be obtained from the exchange list in the console.
factory.setUri	Cluster access address, which can be obtained from the <b>Client Access</b> section on the <b>Ba</b> the cluster.

	<div><div>Client Access</div><div>Network Information</div><div>Web Console Access Address</div><div>Monitor Instance with Prometheus</div></div> <div><div>Client Access ⓘ</div><table><tr><th>Access Type</th><th>Access Policy</th><th>Public Network Ba...</th><th>Network</th><th>Ope</th></tr><tr><td>VPC Network</td><td>-</td><td>-</td><td>vpc- sub- amqp://10.0.2.7:5672</td><td>Dele</td></tr><tr><td>Public domain name access</td><td>Allowlist:  accesses are denied access by default) <a>Modify</a></td><td>3 Mbps <a>Adjust Configuration</a></td><td>amqp</td><td>Dele</td></tr></table></div>	Access Type	Access Policy	Public Network Ba...	Network	Ope	VPC Network	-	-	vpc- sub- amqp://10.0.2.7:5672	Dele	Public domain name access	Allowlist: accesses are denied access by default) <a>Modify</a>	3 Mbps <a>Adjust Configuration</a>	amqp	Dele
Access Type	Access Policy	Public Network Ba...	Network	Ope												
VPC Network	-	-	vpc- sub- amqp://10.0.2.7:5672	Dele												
Public domain name access	Allowlist: accesses are denied access by default) <a>Modify</a>	3 Mbps <a>Adjust Configuration</a>	amqp	Dele												
factory.setVirtualHost	Vhost name, which can be obtained from the vhost list in the console.															
factory.setUsername	Enter the name of the user created in the console.															
factory.setPassword	Enter the password of the user created in the console.															

Step 3. Consume messages

Compile and run `MessageConsumer.java` .



```
import com.rabbitmq.client.AMQP;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.DefaultConsumer;
import com.rabbitmq.client.Envelope;
import com.tencent.tdmq.demo.cloud.Constant;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
```



```
/**

 * Message consumer
 */
public class MessageConsumer1 {

/**

 * Queue name
 */
    public static final String QUEUE_NAME = "queue_name";

/**

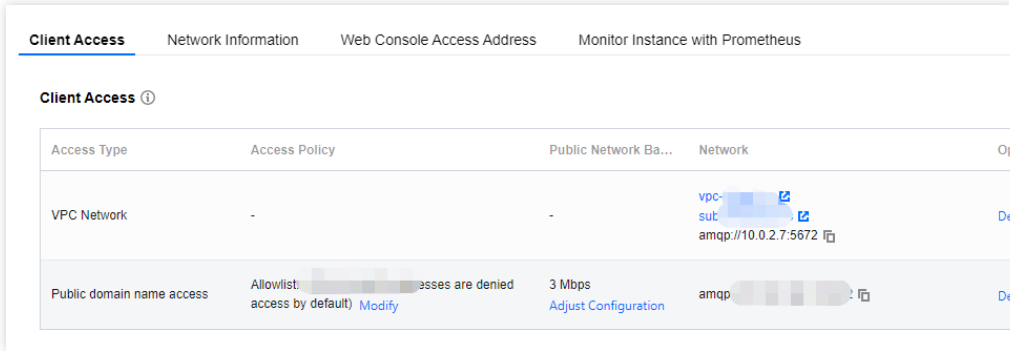
 * Exchange name
 */
    private static final String EXCHANGE_NAME = "exchange_name";

    public static void main(String[] args) throws Exception {
        // Connection factory
        ConnectionFactory factory = new ConnectionFactory();
        // Set the service address (replace with the access point address copied in
        factory.setUri("amqp://***");
        // Set the vhost (copy the vhost name in the open-source RabbitMQ console)
        factory.setVirtualHost(VHOST_NAME);
        // Set the username (use the username in the permission configuration of the
        factory.setUsername(USERNAME);
        // Set the password (use the user key)
        factory.setPassword("****");
        // Get the connection address
        Connection connection = factory.newConnection();
        // Establish a channel
        Channel channel = connection.createChannel();
        // Bind the message exchange
        channel.exchangeDeclare(EXCHANGE_NAME, "fanout");
        // Declare the queue message
        channel.queueDeclare(QUEUE_NAME, true, false, false, null);
        // Bind the message exchange (`EXCHANGE_NAME` must exist in the TDMQ for Rab
        channel.queueBind(QUEUE_NAME, EXCHANGE_NAME, "");
        System.out.println(" [Consumer1] Waiting for messages.");
        // Subscribe to the message
        channel.basicConsume(QUEUE_NAME, false, "ConsumerTag", new DefaultConsumer(c
            @Override
            public void handleDelivery(String consumerTag, Envelope envelope,
                                     AMQP.BasicProperties properties, byte[] body)
                throws IOException {
                // Received message for business logic processing
            }
        });
    }
}
```

```

        System.out.println("Received: " + new String(body, StandardCharsets.
            channel.basicAck(envelope.getDeliveryTag(), false);
    }
}
}
}

```

Parameter	Description
QUEUE_NAME	Queue name, which can be obtained from the queue list in the console.
EXCHANGE_NAME	Exchange name, which can be obtained from the exchange list in the console.
factory.setUri	<p>Cluster access address, which can be obtained from the <b>Client Access</b> section on the <b>Basic Information</b> page of the cluster.</p> 
factory.setVirtualHost	Vhost name, which can be obtained from the vhost list in the console.
factory.setUsername	Enter the name of the user created in the console.
factory.setPassword	Enter the password of the user created in the console.